Comma restoration using constituency information

Proceedings of HLT-NAACL 2003
Main Papers , pp. 142-148
Edmonton, May-June 2003

Comma Restoration Using Constituency Information

Stuart M. Shieber
Harvard University
shieber@deas.harvard.edu

Xiaopeng Tao
Harvard University
xptao@deas.harvard.edu

Abstract

Automatic restoration of punctuation from unpunctuated text has application in improving the fluency and applicability of speech recognition systems. We explore the possibility that syntactic information can be used to improve the performance of an HMM-based system for restoring punctuation (specifically, commas) in text. Our best methods reduce sentence error rate substantially — by some 20%, with an additional 8% reduction possible given improvements in extraction of the requisite syntactic information.

1 Motivation

The move from isolated word to connected speech recognition engendered a qualitative improvement in the naturalness of users' interactions with speech transcription systems, sufficient even to make up in user satisfaction for some modest increase in error rate. Nonetheless, such systems still retain an important source of unnaturalness in dictation, the requirement to utter all punctuation explicitly. In order to free the user from this burden, a transcription system would have to reconstruct the punctuation from the word sequence. For certain applications — for instance, transcription of naturally occurring speech not originally targeted to a speech recognizer (as broadcast audio) — there is no alternative to performing reconstruction of punctuation.

Reconstruction of different punctuation marks is likely to respond to different techniques. Reconstruction of periods, question marks, and exclamation marks, for instance, is in large part the problem of sentence boundary detection. In this paper, we address the problem of comma restoration. The published literature on intrasentence punctuation restoration is quite limited, the state of the art represented by Beeferman, Berger, and Lafferty's CYBERPUNC system, which we review in Section 2, and reimplement as a baseline for our own experiments. (See Section 5 for discussion of related work.)

The CYBERPUNC system uses a simple HMM with trigram probabilities to model the comma restoration problem. It is trained on fully punctuated text, and then tested for precision and recall in reconstructing commas in text that has had them removed. Our replication of the trigram-based method yields a sentence accuracy of 47%.

However, the role of the comma in text is closely related to syntactic constituency. Nunberg (1990) describes two main classes of comma: the delimiter comma, which is used to mark off a constituent, and the separator comma, which is inserted between conjoined elements with or without a conjunction. In both cases, one expects to see commas at the beginning or end of constituents, rather than in the middle. But this type of correlation is difficult to model with a flat model such as an HMM. For this reason, we explore here the use of syntactic constituency information for the purpose of comma restoration. We show that even very rarefied amounts of syntactic information can dramatically improve comma restoration performance; our best method accurately restores 58% of sentences. Furthermore, even approximate syntactic information provides significant improvement.

There is, of course, great variation in appropriate punctuation of a single word stream.[1] For this reason, independent human annotators consider only about 86% of the sentences in the test set to be correct with respect to comma placement (Beeferman et al., 1998). Thus, a move from 47% to 58% is a quite substantial improvement, essentially a reduction in sentence error rate of some 30%.

[1] In an old unattributed joke, an English professor asks some students to punctuate the word sequence "Woman without her man is nothing". The male students preferred "Woman, without her man, is nothing." whereas the female proposed "Woman! Without her, man is nothing." No, it's not funny, but it does make the point.

Digression: What's Statistical Parsing Good For?

There has been a tremendous amount of research since the early 1990's on the problem of parsing using statistical models and evaluated by statistical measures such as crossing brackets rate. Statistical parsing, like language modeling, is presumably of interest not in and of itself but rather by virtue of its contribution to some end-user application. In the case of language modeling, speech recognition is the leading exemplar among a large set of end-user natural-language-processing applications that benefit from the technology. Further, the statistical figures of merit are appropriate just insofar as they vary more or less continuously and monotonically with the performance of the end-user application. Again, in the case of language modeling, speech recognition error rate is generally acknowledged to improve in direct relation to reduction in cross-entropy of the language model employed.

For statistical parsing, it is much more difficult to say what applications actually benefit from this component technology in the sense that incremental improvements to the technology as measured by the statistical figures of merit provide incremental benefit to the application. The leading argument for parsing a sentence is that this establishes the structure upon which semantic interpretation can be performed. But it is hard to imagine in what sense an 85% correct parse is better than an 80% correct parse, as the semantic interpretation generated off of each is likely to be wrong. Barring a sensible notion of "partially correct interpretation" and an end-user application in which a partially correct interpretation is partially as good as a fully correct one, we would not expect statistical parsers to be useful for end user applications based on sentence interpretation.[2] In fact, to the authors' knowledge, the comma restoration results presented here are the first instance of an end-user application that bears on its face this crucial property, that incremental improvement on statistical parsing provides incremental improvement in the application.

2 The Trigram Baseline

As a baseline, we replicated the three-state HMM method of Beeferman et al. (1998). In this section, we describe that method, which we use as the basis for our extensions.

The input to comma restoration is a sentence $x = x_1 \ldots x_n$ of words and punctuation but no commas. We would like to generate a restored string $y = y_1 \ldots y_{n+c}$, which is the string x with c commas inserted. The selected y should maximize conformance with a simple trigram model:

$$y^* = \text{argmax}_y \prod_{i=1}^{n+c} p(y_i \mid y_{i-2} y_{i-1})$$

We take the string x to be the observed output of an HMM with three states and transition probabilities dependent on output; the states encode the position of commas in a reconstructed string. Figure 1 depicts the automaton. The start state (1) corresponds to having seen a word with no prior or following comma, state (2) a word with a following comma, and state (3) a word with a prior but no following comma. It is easy to see that a path through the automaton traverses a string y with probability $\prod_{i=1}^{n+c} p(y_i \mid y_{i-2} y_{i-1})$. The decoded string y^* can therefore be computed by Viterbi decoding.[3]

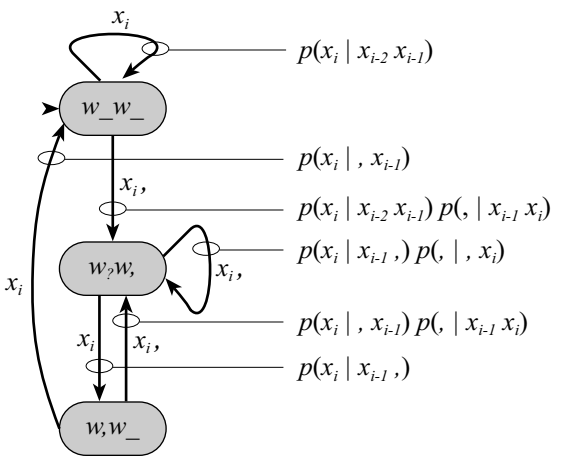


Figure 1: Three-state HMM for decoding a comma-reduced string $x_1 \cdots x_n$ to its comma-restored form. Transitions are labeled with their position-dependent probabilities.

This method requires a trigram language model $p()$. We train this language model on sections 02–22 of the Penn Treebank Wall Street Journal data (WSJ)[4], comprising about 36,000 sentences. The CMU Statistical Language Modeling Toolkit (Clarkson and Rosenfeld, 1997) was used to generate the model. Katz smoothing was used to incorporate lower-order models. The model

[2] We might expect nonstatistical parsers also not to be useful, but for a different reason, their fragility. Rather than delivering partially correct results, they partially deliver correct results. But that is a different issue.

[3] As it turns out, the same computation can be done using a two-state model. This automaton does not, however, lend itself as easily to extensions.

[4] For consistency, we use the version of the Wall Street Journal data that was used by Beeferman et al. (1998) for their CYBERPUNC experiments. This comprises sections 02–23 of the Wall Street Journal (the last of these being used as test data) with minor variations from the Treebank version, for instance, a small number of missing sentences and some variation in the tags. Runs of the experiments below using the Treebank versions of the data yield essentially identical results.

was then tested on the approximately 2300 sentences of WSJ Section 23. Precision of the comma restoration was 71.1% and recall 55.2%. F-measure, calculated as $2PR/(P + R)$, where P is precision and R recall, is 62.2%. Overall 96.3% of all comma placement decisions were made correctly, a metric we refer to as *token accuracy*. *Sentence accuracy*, the percentage of sentences correctly restored, was 47.0%. (These results are presented as model 1 in Table 1.) This is the baseline against which we evaluate our alternative comma restoration models.

Beeferman et al. present an alternative trigram model, which computes the following:

$$y^* = \text{argmax}_y \prod_{i=1}^{n+c} p(y_i \mid y_{i-2}y_{i-1}) \\ (1 - p(, \mid y_{i-2}y_{i-1}))^{\delta(y_i)}$$

where

$$\delta(y_i) = \begin{cases} 0 & y_i =, \\ 1 & otherwise \end{cases}$$

That is, an additional penalty is assessed for not placing a comma at a given position. By penalizing omission of a comma between two words, the model implicitly rewards commas; we would therefore expect higher recall and correspondingly lower precision.[5] In fact, the method with the omission penalty (model 2 in Table 1), does have higher recall and lower precision, essentially identical F-measure, but lower sentence accuracy. Henceforth, the models described here do not use an omission penalty.

3 Commas and Constituency

Insofar as commas are used as separators or delimiters, we should see correlation of comma position with constituent structure of sentences. A simple test reveals that this is so. We define the start count sc_i of a string position i as the number of constituents whose left boundary is at i. The end count ec_i is defined analogously. For example, in Figure 2, sc_0 is 4, as the constituents labeled JJ, NPB, S, and S start there; ec_0 is 0. We compute the end count for positions that have a comma by first dropping the comma from the tree. Thus, at position 5, sc_5 is 2 (constituents DT, NPB) and ec_5 is 4 (constituents JJ, ADJP, VP, S).

We expect to find that the distributions of sc and ec for positions in which a comma is inserted should differ from those in which no comma appears. Figure 3 reveals that this intuition is correct. The charts show the percentage of string positions with each possible value of sc (resp. ec) for those positions with commas and those

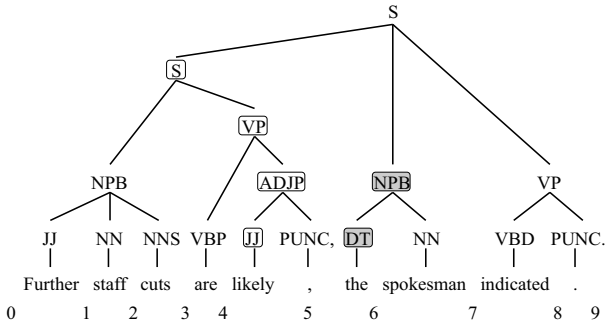


Figure 2: Sample tree, showing computation of sc and ec values. The four constituents leading to $ec_5 = 4$ are shown circled, and the two leading to $sc_5 = 2$ are shown circled and shaded.

without. We draw the data again from sections 02–22 of the Wall Street Journal, using as the specification for the constituency of sentences the parses for these sentences from the Penn Treebank. The distributions are quite different, hinting at an opportunity for improved comma restoration.

The ec distribution is especially well differentiated, with a cross-over point at about 2 constituents. We can add this kind of information, a single bit specifying an ec value of k or greater (call it $\widehat{ec_i}$), to the language model, as follows. We replace $p(y_i \mid y_{i-2}y_{i-1})$ with the probability $p(y_i \mid y_{i-2}y_{i-1}\widehat{ec_i})$. We smooth the model using lower order models $p(y_i \mid y_{i-1}\widehat{ec_i})$, $p(y_i \mid \widehat{ec_i})$, $p(y_i)$.[6] These distributions can be estimated from the training data directly, and smoothed appropriately.

Adding just this one bit of information provides significant improvement to comma restoration performance. As it turns out, a k value of 3 turns out to maximize performance.[7] Compared to the baseline, F-measure increases to 63.2% and sentence accuracy to 52.3%. This experiment shows that constituency information, even in rarefied form, can provide significant performance improvement in comma restoration. (Figure 1 lists performance figures as model 3.)

Of course, this result does not provide a practical algorithm for comma restoration, as it is based on a probabilistic model that requires data from a manually constructed parse for the sentence to be restored. To make the method practical, we might replace the Treebank parse with a statistically generated parse. In the sequel, we use Collins's statistical parser (Collins, 1997) as our canonical automated approximation of the Treebank. We can train a similar model, but using ec values extracted from

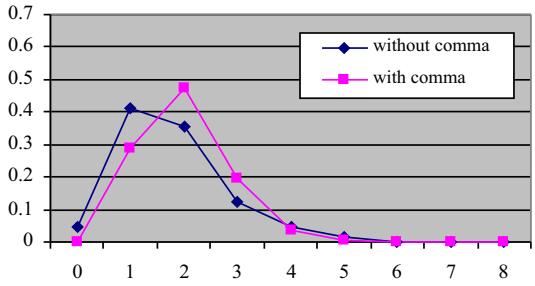
[5] Counterintuitively, Beeferman et al. (1998) come to the opposite expectation, and their reported results bear out their intuition. We have no explanation for this disparity with our results.

[6] Alternative backoff paths, for instance backing off first to $p(y_i \mid y_{i-2}y_{i-1})$, exhibit inferior performance.

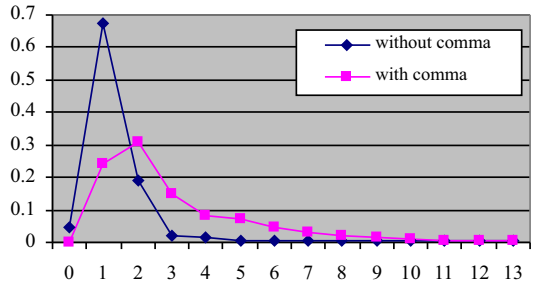
[7] With $k = 2$ (model 4), precision drops precipitously to 60.4%, recall stays roughly the same at 66.4%.

Table 1: Performance of the various comma restoration models described in this paper.

Info sources								**Training**			**Testing**								
trigram	insertion penalty	word class	ec, threshold = 2	ec, threshold = 3	ec, no threshold	stemmer	Treebank	Collins parse, commas	Collins parse, no commas	Treebank	Collins parse, commas	Collins parse, no commas	**Model number**	precision	recall	F-measure	token accuracy	sentence accuracy	reduction in sentence "error"
•							•			•			1	.711	.552	.621	.963	.470	.000
•	•												2	.684	.576	.625	.962	.457	-.033
•		•					•			•				.834	.511	.634	.967	.514	.113
•						•	•			•				.709	.559	.625	.963	.464	-.014
•				•			•			•			3	.752	.627	.683	.968	.523	.135
•			•				•			•			4	.604	.664	.633	.958	.435	-.091
•		•		•			•			•				.863	.563	.681	.971	.555	.217
•					•		•			•			8	.671	.780	.721	.967	.508	.097
•		•			•		•			•			10	.796	.704	.748	.974	.579	**.280**
•		•			•	•	•			•			11	.791	.711	.749	.974	.576	.271
•				•					•			•	5	.714	.588	.645	.964	.489	.049
•			•						•			•		.733	.557	.633	.964	.489	.048
•		•		•					•			•		.851	.532	.655	.969	.534	.164
•					•				•			•	9	.657	.674	.666	.963	.476	.015
•		•			•				•			•	12	.797	.626	.701	.970	.549	**.204**
•		•			•	•			•			•		.791	.631	.702	.970	.544	.190
•				•				•				•	6	.714	.581	.641	.964	.486	.041
•				•				•			•		7	.738	.609	.668	.967	.507	.095
•			•					•			•			.746	.602	.666	.967	.503	.085
•		•		•				•			•			.859	.556	.675	.970	.550	.205
•					•			•			•			.681	.728	.704	.966	.501	.080
•		•			•			•			•			.811	.672	.735	.973	.571	.260
•		•			•	•		•			•			.805	.677	.735	.973	.570	.256



(a)



(b)

Figure 3: Differential pattern of constituent starts and ends for string positions with and without commas. Chart (a) shows the percentage of constituents with various values of sc (number of constituents starting at the position) for string positions with commas (square points) and without (diamond points). Chart (b) shows the corresponding pattern for values of ec (number of constituents ending).

Collins parses of the training data, and use the model to restore commas on a test sentence again using ec values from the Collins parse of the test datum. This model, listed as model 5 in Table 1, has an F-measure of 64.5%, better than the pure trigram model (62.2%), but not as good as the oracular Treebank-trained model (68.4%). The other metrics show similar relative orderings.

In this model, since the test sentence has no commas initially, we want to train the model on the parses of sentences that have had commas removed, so that the model is being applied to data as similar as possible to that on which it was trained. We would expect, and experiments verify (model 6), that training on the parses with commas retained yields inferior performance (in particular, F-measure of 64.1% and sentence accuracy of 48.6%). Again consistent with expectations, if we could clairvoyantly know the value of $\widehat{ec_i}$ based on a Collins parse of the test sentence with the commas that we are trying to restore (model 7), performance is improved over model 5; F-measure rises to 66.8%.

The steady rise in performance from models 6 to 5 to 7 to 3 exactly tracks the improved nature of the syntactic information available to the system. As the quality of the syntactic information better approximates ground truth, our ability to restore commas gradually and monotonically improves.

4 Using More Detailed Syntactic Information

4.1 Using full end count information

The examples above show that even a tiny amount of syntactic information can have a substantive advantage for comma restoration. In order to use more information, we might imagine using values of ec directly, rather than thresholding. However, this quickly leads to data sparsity problems. To remedy this, we assume independence between the bigram in the conditioning context and the syntactic information, that is, we take

$$p(y_i \mid y_{i-2}y_{i-1}ec_i) \approx \frac{p(y_i \mid y_{i-2}y_{i-1})p(y_i \mid y_{i-1}ec_i)}{p(y_i)}$$

This model[8] (model 8) has an F-measure of 72.1% due to a substantial increase in recall, demonstrating that the increased articulation in the syntactic information available provides a concomitant benefit. Although the sentence accuracy is slightly less than that with thresholded ec, we will show in a later section that this model combines well with other modifications to generate further

[8] We back off the first term in the approximation as before, and the second to $p(y_i \mid y_{i-1})$.

improvements.[9]

4.2 Using part of speech

Additional information from the parse can be useful in predicting comma location. In this section, we incorporate part of speech information into the model, generating model 10. We estimate the joint probability of each word x_i and its part of speech X_i as follows:

$$p(x_i, X_i \mid x_{i-2}, X_{i-2}, x_{i-1}, X_{i-1}, ec) \approx p(x_i \mid x_{i-2}x_{i-1}ec)p(X_i \mid X_{i-2}X_{i-1})$$

The first term is computed as in model 8, the second backing off to bigram and unigram models. Adding a part of speech model in this way provides a further improvement in performance. F-measure improves to 74.8%, sentence accuracy to 57.9%, a 28% improvement over the baseline.

These models (8 and 10), like model 3, assumed availability of the Treebank parse and part of speech tags. Using the Collins-parse-generated parses still shows improvement over the corresponding model 5: an F-measure of 70.1% and sentence accuracy of 54.9%, twice the improvement over the baseline as exhibited by model 5.

5 Related Work

We compare our comma restoration methods to those of Beeferman et al. (1998), as their results use only textual information to predict punctuation. Several researchers have shown prosodic information to be useful in predicting punctuation (Christensen et al., 2001; Kim and Woodland, 2001) (along with related phenomena such as disfluencies and overlapping speech (Shriberg et al., 2001)). These studies, typically based on augmenting a Markovian language model with duration or other prosodic cues as conditioning features, show that prosody information is orthogonal to language model information; combined models outperform models based on each type of information separately. We would expect therefore, that our techniques would similarly benefit from the addition of prosodic information.

In the introduction, we mentioned the problem of sentence boundary detection, which is related to the punctuation reconstruction problem especially with regard to predicting sentence boundary punctuation such as periods, question marks, and exclamation marks. (This problem is distinct from the problem of sentence boundary disambiguation, where punctuation is provided, but the categorization of the punctuation as to whether or not it marks a sentence boundary is at issue (Palmer and Hearst, 1994; Reynar and Ratnaparkhi, 1997).) Stolcke and Shriberg (1996) used HMMs for the related problem of *linguistic segmentation* of text, where the segments corresponded to sentences and other self-contained units such as disfluencies and interjections. They argue that a linguistic segmentation is useful for improving the performance and utility of language models and speech recognizers. Like the present work, they segment clean text rather than automatically transcribed speech. Stevenson and Gaizauskas (Stevenson and Gaizauskas, 2000) and Goto and Renals (Gotoh and Renals, 2000) address the sentence boundary detection problem directly, again using lexical and, in the latter, prosodic cues.

6 Future Work and Conclusion

The experiments reported here — like much of the previous work in comma restoration (Beeferman et al., 1998) and sentence boundary disambiguation and restoration (Stolcke and Shriberg, 1996; Shriberg et al., 2001; Gotoh and Renals, 2000; Stevenson and Gaizauskas, 2000) (though not all (Christensen et al., 2001; Stolcke et al., 1998; Kim and Woodland, 2001)) — assume an ideal reference transcription of the text. The performance of the method on automatically transcribed speech with its concomitant error remains to be determined. A hopeful sign is the work of Kim and Woodland (Kim and Woodland, 2001) on punctuation reconstruction using prosodic information. The performance of their system drops from an F-measure of 78% on reference transcriptions to 44% on automatically transcribed speech at a word error rate of some 20%. Nonetheless, prosodic features were still useful in improving the reconstructed punctuation even in the automatically transcribed case.

The simple HMM model that we inherit from earlier work dramatically limits the features of the parse that we can easily appeal to in predicting comma locations. Many alternatives suggest themselves to expand the options, including maximum entropy models, which have been previously successfully applied to, inter alia, sentence boundary detection (Reynar and Ratnaparkhi, 1997), and transformation-based learning, as used in part-of-speech tagging and statistical parsing applications (Brill, 1995).

In addition, all of the methods above are essentially nonhierarchical, based as they are on HMMs. An alternative approach would use the statistical parsing model itself as a model of comma placement, that is, to select the comma placement for a string such that the resulting reconstructed string has maximum likelihood under the statistical parsing model. This approach has the benefit that the ramifications of comma placement on all aspects of the syntactic structure are explored, but the disadvantage that the longer distance lexical relationships found in a trigram model are eliminated.

[9] An alternative method of resolving the data sparsity issues is to back off the model $p(y_i \mid y_{i-2}y_{i-1}ec_i)$, for instance to $p(y_i \mid y_{i-2}y_{i-1})$ or to $p(y_i \mid y_{i-1}ec_i)$. Both of these perform less well than the approximation in model 8.

Nonetheless, even under these severe constraints and using quite simple features distilled from the parse, we can reduce sentence error by 20%, with the potential of another 8% if statistical parsers were to approach Treebank quality. As such, comma restoration may stand as the first end-user application that benefits from statistical parsing technology smoothly and incrementally. Finally, our methods use features that are orthogonal to the prosodic features that other researchers have explored. They therefore have the potential to combine well with prosodic methods to achieve further improvements.

Acknowledgments

Partial support for the work reported in this paper was provided by the National Science Foundation under grant number IRI 9712068.

We are indebted to Douglas Beeferman for making available his expertise and large portions of the code and data for replicating the CYBERPUNC experiments.

The first author would like to express his appreciation to Ivan Sag and the Center for the Study of Language and Information, Stanford, California and to Oliviero Stock and the Centro per la Ricerca Scientifica e Tecnologica, Trento, Italy, for space and support for this work during spring and summer of 2002.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1998. CYBERPUNC: A lightweight punctuation annotation system for speech. In *Proceeding as of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 689–692, Seattle, WA.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

Heidi Christensen, Yoshihiko Gotoh, and Steve Renals. 2001. Punctuation annotation using statistical prosody models. In *Proceedings of the 2001 ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, Red Bank, NJ, October 22–24. International Speech Communication Association.

Philip Clarkson and Ronald Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of Eurospeech '97*, pages 2707–2710, Rhodes, Greece, 22–25 September.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, 7–11 July.

Yoshihiko Gotoh and Steve Renals. 2000. Sentence boundary detection in broadcast speech transcripts. In *Proceedings of the ISCA Workshop on Automatic Speech Recognition: Challenges for the New Millennium (ASR-2000)*, Paris, France, 18–20 September. International Speech Communication Association.

Ji-Hwan Kim and P. C. Woodland. 2001. The use of prosody in a combined system for punctuation generation and speech recognition. In *Proceedings of Eurospeech '01*, pages 2757–2760, Aalborg, Denmark, September 3–7.

Geoffrey Nunberg. 1990. *The Linguistics of Punctuation*. CSLI Publications, Stanford, CA.

David D. Palmer and Marti A. Hearst. 1994. Adaptive sentence boundary disambiguation. In *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing*, pages 78–83, Stuttgart, Germany, 13–15 October. Morgan Kaufmann.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, DC, 31 March–3 April.

Elizabeth Shriberg, Andreas Stolcke, and Don Baron. 2001. Can prosody aid the automatic processing of multi-party meetings? Evidence from predicting punctuation, disfluencies, and overlapping speech. In *Proceedings of the 2001 ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, Red Bank, NJ, October 22–24. International Speech Communication Association.

Mark Stevenson and Robert Gaizauskas. 2000. Experiments on sentence boundary detection. In *Proceedings of the Sixth Conferernce on Applied Natural Language Processing and the First Conference of the North American Chapter of the Association for Computational Linguistics*, pages 24–30, Seattle, WA, April.

Andreas Stolcke and Elizabeth Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In H. T. Bunnell and W. Idsardi, editors, *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1005–1008, Philadelphia, PA, 3–6 October.

Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Mari Ostendorf, Dilek Hakkani, Madelaine Plauche, Gökhan Tür, and Yu Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proceedings of the International Conference on Spoken Language Processing*, volume 5, pages 2247–2250, Sydney, Australia, 30 November–4 December.